Assurance

Amit Konar
Math and Computer Sc., UMSL

Assurance and Trust?

Vendors frequently use the term “secure” in product names and product literature to refer to products and systems that have some security included in their design and implementation.

Providing security requirements and functionality may not be sufficient to engender trust in the system.

Intuitively, **trust is a belief or desire that a computer entity will do what it should** to protect resources and be safe from attack.

Definition: Trustworthy

An entity is **trustworthy** if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements. Trust is a measure of trustworthiness, relying on the evidence provided.

To determine trustworthiness, we focus on methodologies and metrics that allow us to measure the degree of confidence that we can place in the entity under consideration.

Definition: Security Assurance

Security assurance, or simply **assurance**, is confidence that an entity meets its security requirements, based on specific evidence provided by the application of assurance techniques.

A related term, **information assurance**, refers to the ability to access information and preserve the quality and security of that information.

It differs from security assurance, because the focus is on the threats to information and the mechanisms used to protect information, and not on the correctness, consistency or completeness of the requirements and implementation of those mechanisms.

Definition: Trusted System

A **trusted system** is a system that has been shown to meet well-defined requirements under an evaluation by a credible body of experts who are certified to assign trust ratings to evaluated products and systems.

Need for Assurance

Accidental/ unintentional failures of computer systems, as well as intentional compromises of security mechanisms, can lead to security failures. Neumann describes 9 types of problem sources in computer systems.

1. **Requirement definitions, omissions, and mistakes**
2. **System design flaws**
3. **Hardware implementation flaws, such as wiring or chip faults**
4. **Software implementation errors, program bugs and compiler bugs**
5. **System use and operation errors and inadvertent mistakes**
6. **Willful system misuse**
7. **Hardwired communication or other equipment malfunction**
8. **Environmental problems**
9. **Evolution, maintenance, faulty upgrades, and decommissions**

Examples illustrating the need for Assurance

Example 1: The space shuttle Challenger exploded on January 28, 1986, killing everyone on board.

An essential failure was a decision to take shortcuts to meet an accelerated launch schedule. Among other steps, several sensors were removed from the booster rockets. The sensors might have enabled analysts to detect that the cold weather was affecting the booster rockets adversely, and to delay the launch. **Better Assurance techniques might have detected the possible effects of removing the sensors**

Example 2: The Radiation Overdose Death Problem

Three patients died from a radiation overdose attributed to a Therac 25 computer-based electron accelerator radiation therapy system.

The flaws in the system resulted from two flaws in the design of the system's software and the removal of a hardware safety interlock. **Assurance techniques would have detected the removal of the interlock.**

Example 3: The Three Mile Island Nuclear Failure

Although the most significant root cause of the Three Mile Island Nuclear failure was a hardware problem (nonstandard instruments were used to measure core temperature), **design and software problems contributed significantly.**

When the temperature rose very high, the system printed a string of question marks rather than the measured temperature. In addition, the intended, rather than the actual valve settings were displayed. Assurance techniques would have detected these software flaws.

Example 4: The Bell V22 Osprey helicopter failure

The Bell V22 Osprey is a high technology helicopter. After a fifth Osprey had crashed, an analysis traced the cause to a failure to correct for malfunctioning components.

The Osprey implemented a majority-voting algorithm, and the cross-wiring of two roll-state sensors allowed two faulty components to outvote the third, the correctly functioning component. Although assurance techniques might not have prevented the incorrect voting, they would have emphasized the results that could have occurred if faulty components overrode correctly functioning components.

Example 5: Trigonometric function failures in Intel's 486 Chip

When bugs were found in the trigonometric functions of the Intel's 486 chip, Intel's public reputation was damaged, and replacing the chips cost Intel time and money.

As a result, Intel began using high **assurance methods** to verify the correctness of requirements in their chip design.